LLM Classification Finetuning

Matthew Band,[1] Mohamad Ali Jarkas[1] and Henrique Jongh[1]

[1]Electrical and Computer Engineering, McGill University, 845 Sherbrooke Street West, H3A 0G3, Quebec, Canada

Abstract

This paper contributes to the LMSYS LLM Classification Fine-Tuning Kaggle competition and related research. Our objective is to develop a model capable of determining which of two responses, generated by different language models, is preferred given a shared prompt. We present five architectures: a single BERT model, a parallel BERT model, their respective augmented variants fine-tuned using LLM-generated preference annotations, and a decoder-only model using Meta's LLaMA 3.1 8B. The LLaMA model achieved the lowest log loss on the competition test set 0.98, suggesting a positive correlation between model capacity and classification performance.

Key words: Human Preference, LLM Alignment, Chatbot Arena, Transformers, Encoders

Introduction

Human preference modeling represents a core technical challenge in large language model (LLM) development. As LLMs continue to integrate into various aspects of society, aligning them with human values becomes increasingly crucial, to ensure these systems respect societal norms and ethical standards while mitigating potentially harmful or inappropriate responses. The challenge lies in effectively capturing the multidimensional nature of human judgment, as traditional preference modeling often reduces complex evaluations to simplified binary signals, failing to represent the nuanced criteria users apply when assessing AI responses.

Human preference is an inherently complex and subjective domain. What constitutes a "good" or "preferred" response varies widely across individuals, cultures, contexts, and tasks. This subjectivity presents a significant challenge when training models to predict human preferences, as they are determined by a complex evaluation process that humans perform intuitively.

Historically, encoder-only transformer architectures like BERT have dominated natural language classification tasks due to their bidirectional attention mechanisms and efficient parameter usage. These models excel at contextual understanding and have established strong benchmarks across various domains, such as sentiment analysis. However, recent research demonstrates the emerging potential of decoder-only large language models for classification tasks, despite their traditionally generative focus. These larger models leverage their extensive knowledge corpus to potentially capture more subtle aspects of language, highlighting a growing recognition that complex judgment tasks, particularly those involving human preference prediction, may benefit from the increased capacity and sophisticated representations available in modern LLMs.

This research examines transformer-based approaches to predict human preferences between LLM outputs, moving beyond binary classifications toward multidimensional preference modeling by exploring ways to augment existing datasets with more granular preference dimensions beyond simple binary choices.

Background and Related Work

The field of AI alignment has attempted to formalize aspects of preferred AI behavior through various frameworks. One influential approach introduced in Askell et al. [1] established the HHH framework as guiding principles for LLM development and alignment:

- **Helpful**: Systems should provide useful, relevant information and assistance
- **Honest**: Systems should be truthful and acknowledge limitations
- **Harmless**: Systems should avoid generating harmful, misleading, or unethical content

While the HHH framework provides valuable high-level orientation, our work requires more granular criteria to accurately predict human preference judgments between specific responses. To address this need, we turned to empirically grounded research on preference dimensions.

Li et al. [2] offers a valuable contribution by establishing a detailed taxonomy of preference dimensions based on empirical analysis. Rather than relying on abstract principles, this taxonomy categorizes preference indicators into three primary groups:

- **Basic Properties**: 21 properties covering stylistic and content elements like harmlessness, grammar correctness, friendliness, politeness, clarity, information richness, and novelty.
- **Query-specific Properties**: 5 properties that evaluate how responses address specific aspects of user queries, including clarifying ambiguous intent, showing empathy, satisfying explicit constraints, supporting user stances, and correcting mistakes.
- **Error Detection Properties**: 3 severity levels (minor, moderate, severe) for measuring factual errors, information contradictions, mathematical errors, and code generation problems

This comprehensive taxonomy allows for measuring multiple dimensions that influence human preference judgments, providing a richer framework than abstract principles alone.

LLM Biases vs. Human Preferences

This taxonomy reveals some interesting differences between what humans value and what LLMs have been optimized for through alignment techniques:

Human Preferences:

- Less sensitivity to factual errors, especially minor and moderate ones
- Dislike when models admit limitations, particularly in communication scenarios
- Preference for responses that support subjective stances ("sycophancy")

LLM Preferences:

- Strong emphasis on correctness and error aversion
- Focus on harmlessness and ethical considerations
- Willingness to clarify user intent and correct mistakes

These discrepancies suggest that current alignment techniques may overemphasize certain aspects (like those in HHH) at the expense of other factors that humans actually value in practice. Understanding these differences is crucial for our preference prediction task, as mimicking LLM alignment criteria may not accurately reflect real human preferences.

LLMs as Preference Evaluators

While not perfectly aligned with human values, LLMs can still reliably evaluate responses across most taxonomy categories with reasonable accuracy [2]. This reliability makes LLMs promising tools for dataset augmentation, as they can generate granular scores across multiple preference dimensions to create richer representations of prompt-response pairs. Using LLMs as evaluators allows for efficient scaling of training data without requiring extensive human annotation while still capturing many of the nuanced dimensions that influence human preference judgments.

BERT for Text Classification

Unlike traditional NLP tasks with objective ground truth (such as question answering or machine translation), predicting human preferences requires modeling implicit evaluation criteria that vary across individuals and contexts.

Bidirectional Encoder Representations from Transformers (BERT) has emerged as a powerful architecture for text classification tasks due to its ability to capture bidirectional contextual representations. Unlike earlier unidirectional models, BERT pre-training enables the model to consider both left and right context simultaneously, creating rich semantic representations of text. This bidirectional context modeling is particularly valuable for classification tasks where understanding the full context is essential, providing a convenient aggregated representation for classification decisions. Finally, BERT's transfer learning capabilities allow models to leverage general language understanding while fine-tuning to different tasks, such as multi-class classification.

Chatbot Arena and Human Preference Evaluation

Chatbot Arena represents one of the most influential benchmarking platforms for large language models based entirely on human feedback [2]. Developed by LMSYS Org, this platform implements a large-scale, crowd-sourced evaluation methodology that directly captures human preferences between language model outputs.

The Arena employs an Elo rating system, similar to that used in chess rankings, to establish a relative performance hierarchy among competing models. Through this system, each model maintains a dynamic rating score that evolves based on the outcomes of pairwise comparisons. When a model "wins" in a comparison (i.e., its response is preferred by a human evaluator), it gains rating points, while the "losing" model loses points. The magnitude of these adjustments depends on the relative ratings of the competitors.

The evaluation process follows a blind testing protocol where users interact with two anonymous language models simultaneously. For each user query, both models generate responses, and the user selects which response they prefer (or indicates a tie). This approach allows for capturing human preferences directly rather than using proxy metrics.

Kaggle Competition: LLM Classification Fine-Tuning

The data and evaluation framework for this research comes from Kaggle's "LLM Classification Fine-Tuning" competition [3]. This competition addresses the challenge of predicting human preferences between pairs of LLM responses to the same prompt. Participants are provided with a dataset from Chatbot Arena, where users submit prompts to two anonymous chatbots and indicate their preference between the responses.

The validity of this data has been established by the dataset creators [4], providing a consistent and valid representation of human preferences from the platform's user population at the time of data collection.

The competition's evaluation metric is logarithmic loss, with submissions ranked based on their performance on a hidden test set. Importantly, the competition requires models to output probability distributions across three classes (Winner A, Winner B, or Tie) rather than simple binary decisions [3].

This competition framework provides a standardized evaluation environment with real-world human preference data, allowing for direct comparison between different modeling approaches. The structure of the competition (requiring models to compare paired responses to the same prompt) mirrors how human evaluators naturally assess AI systems, making it an ideal testbed for human preference modeling research.

Problem Statement

Building on the insights from prior work on human preference modeling, we focus on developing a classification system for the preference prediction challenge through multiple model architectures.

Task Definition

Formally, the task can be defined as follows: Given a user prompt P and two model-generated responses R_A and R_B, our objective is to train a classifier $f(P, R_A, R_B)$ that outputs a probability distribution over three possible outcomes:

- Winner A: Response A is preferred
- Winner B: Response B is preferred
- Tie: Both responses are considered equally valuable

Evaluation Metric

The classifier should minimize log loss (cross-entropy) between its predictions and actual human preferences, defined as:

$$\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{3} y_{ij}\log(p_{ij})$$

Where N is the number of samples, y_{ij} is a binary indicator of whether class j is the correct classification for observation i, and p_{ij} is the predicted probability that observation i belongs to class j. Lower log loss values indicate better prediction accuracy.

Constraints and Requirements

The solution must satisfy several critical constraints:

1. **Computational Efficiency**: The model must train within the limited computational resources available (2×T4 GPUs and 1xTPU with 30 and 20 hours/week respectively of compute time).
2. **Context Window Utilization**: The model must effectively handle the full prompt and response texts, which often exceed standard transformer context windows (512 tokens) found in models like BERT.
3. **Generalizability**: The model should generalize to new prompts and responses beyond the training data, validated by final performance metric on the held-out test set.
4. **Performance Target**: Achieve a log loss below 1.0 on the unseen test set, a self-imposed benchmark that would position our solution competitively within the top 25 submissions on the Kaggle leaderboard.

These constraints guided our model architecture decisions, training strategies, and data augmentation approaches described in the following sections.

Methodology

ChatBot Arena Dataset

As previously mentioned, the dataset used in this study comes from Kaggle's *LLM Classification Fine Tuning* challenge. Figure 1 provides a visual example of how the prompt-response pairs appear on the platform.

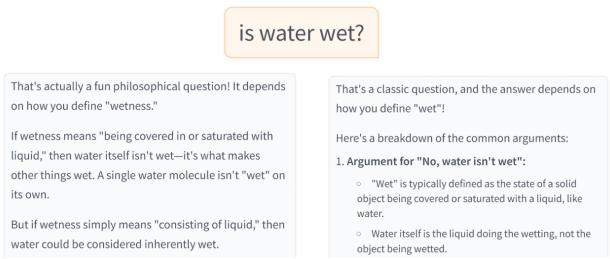

Figure 1. ChatBot Arena Example Prompt and Responses[5]

The dataset consists of 57,486 rows, with each row containing the user *prompt*, the two model responses (labeled as *response_a* and *response_b*), and the human preference judgment encoded as a one-hot vector across three classes: *winner_a*, *winner_b*, or *tie*. Figure 2 displays a sample row from the dataset, illustrating its structure.

prompt	response_a	response_b	winner_model_a	winner_model_b	winner_tie
["Is it mora	["The questior	["As an AI, I d	1	0	0

Figure 2. Row of Training Data from the Kaggle dataset [3]

Training, Validation, and Testing Dataset

When it comes to training, validating, and testing a proposed model, we opted to randomly split this given data with 90% for training, and 10% for validation. The competition provides an anonymously sized test set available upon submission to the competition, therefore we opted to use this set and not create a test split from our data. Still, it is important to note that although this approach streamlines comparison with other competitors, it limits our ability to compute metrics such as accuracy, recall, F1 scores, etc.

Data Augmentation

LLM-Based Preference Evaluation Framework

Building on the preference dimensions taxonomy described in the Background section, we developed a framework to augment our training data, leveraging the capabilities of LLMs to evaluate text across multiple dimensions as established in prior research [2].

To ensure consistent and explainable evaluations, we employed a Chain of Draft (CoD) [6] approach, where the model first performs internal reasoning within designated tags, then summarizes its thinking into minimal representations focused on essential observations:

```
Traditional reasoning: "This response provides detailed
background information about the topic, offers multiple
perspectives with supporting evidence, and addresses
potential counterarguments in a balanced manner."

Chain of Draft: "Rich info, multi-perspective,
balanced counterpoints"
```

Implementation and Dataset Generation

Our implementation process followed these key steps:

1. **Prompt Engineering**: We developed structured prompts with five components:
   - Task definition, few-shot examples, detailed grading scale (0-3), property definitions, and structured output format (JSON template).
2. **Batch Processing**: We implemented resumable operations with progress tracking, allowing the pipeline to continue from the last successful point after interruptions.
3. **Robustness Measures**: Protection against API failures, output parsing errors, and consistent handling of edge cases.

The final augmented dataset expanded each original sample with:

- 5 query-specific binary property classifications.
- 18 basic property ratings (0-3) for each response.
- 2 query-aware property ratings (0-3) for each response.
- Associated reasoning for each rating (preserved but not used in current models).

Due to resource constraints, we were not able to extract information from the entire dataset, limiting ourselves to a subset of approximately 4000 queries paired with each A/B response, totaling 8000 samples of augmented which was used to distill knowledge from a 70-billion-parameter reasoning LLM.

Model Architectures

The task is to train a model to predict response preference—a standard text classification problem, similar to sentiment analysis, topic categorization, or spam detection. Traditionally, encoder models like BERT [7] have been widely used for such tasks. However, leader board results on Kaggle show that LLMs currently outperform them, likely due to the increased capacity and improved pre-training techniques.

The selected model will be fine-tuned on input data (prompt, response_a, response_b) and corresponding labels (winner_a, winner_b, tie). The objective function minimized is the log loss (cross-entropy). To enable gradient updates, the model must output logits—not probabilities. During inference, logits are passed through a softmax.

Core Models

BERT is pre-trained on a large text corpus, enabling it to produce rich textual embeddings. Prior research shows BERT performs well when fine-tuned for tasks like question answering, sentiment analysis, and news categorization. Thus, it is a natural starting point for the proposed architecture.[7].

Numerous BERT variants exist, including RoBERTa-large, BERT-base, DistilBERT, BART-large-MNLI, Longformer, and BigBird [8]. For this project, we focused on *BERT-base* and the smaller *DistilBERT*. BERT-base is a widely adopted model, and comparing it with its lighter counterpart should yield interesting results. DistilBERT's reduced parameter count also made it more convenient for rapid iteration and establishing a reliable encoder training pipeline. This leaves room for future exploration of other variants—particularly Longformer, due to its extended context window.

Despite the extensive use of BERT-like models for text classification, a review of top-performing models in the competition revealed a surprising trend: encoder-based transformers were not among the best performers. Instead, LLMs consistently dominate the top rankings of the leaderboard. To investigate this trend, we also included LLaMA 3.1–8B as a representative LLM for comparison.

This is notable because encoder-based models like BERT are specifically designed for text representation and can be adapted for classification with minimal changes. In contrast, LLMs are trained solely for next-token prediction and require prompting or fine-tuning to perform classification.

A brief summary of the chosen architectures is presented on table 1.

Table 1. Core models and their specifications

Core Model Name	Transformer Type	Parameters (million)
DistilBERT	Encoder	66
BERT-base	Encoder	110
llama3.1-8B	Decoder	7,508

Custom Tokenizer for BERT

Transformer-based architectures fundamentally require tokenization. For example, in sentiment analysis with BERT, the input text must first be tokenized before being fed into the model. When using models from the Transformers library, a default tokenizer is typically provided. Still, a modification was required to balance input across the limited 512-token context window of the BERT-base and DistilBERT models. Our dataset showed that single responses could reach up to 7,449 tokens, with 90% of samples being $\leq$1390 tokens. Naively concatenating the prompt and both responses would exceed the 512-token limit, causing the model to truncate input and ignore critical information. Depending on the concatenation order, parts of the prompt or a response could be truncated. Figure 3 illustrates this overflow, where only the prompt and the first response fit within the context window, while the second response is almost completely discarded.

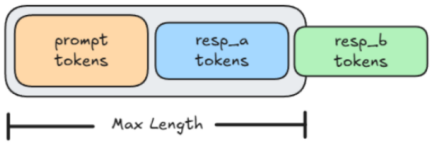

Figure 3. *Overflowed* Context Window representation

To address the context window limitation and better represent the input structure, a custom token allocation scheme was implemented. Instead of naive concatenation, fixed percentage splits were used to allocate space within the 512-token limit. For instance, using ratios like [0.1, 0.45, 0.45] reserves 10% of the space for the prompt and 45% each for *response_a* and *response_b*. While this results in truncating some trailing content, it ensures a balanced representation. Tail truncation was chosen as the strategy, as supported in [9]. Figure 4 illustrates this space allocation strategy.

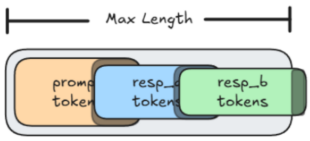

Figure 4. Percentage Space Allocation. Dark shading is truncation

For completeness, the custom tokenizer scheme also includes a classification token `[CLS]` and separation tokens `[SEP]`. The `[CLS]` token is prepended to the input and serves as a summary representation—its final hidden state is used for classification. The `[SEP]` token separates segments in the input. Since BERT is designed to process sentence pairs, `[SEP]` tokens are inserted between the prompt and *response_a*, and between *response_a* and *response_b* to help the model distinguish between segments.

Single BERT (baseline) **(Model 1/5)**

The simplest proposed model architecture is the Single BERT architecture. This approach uses one BERT model to perform *winner_a*, *winner_b*, and *tie* classification. As only a single encoder, this means that *prompt*, *response_a*, and *response_b* must be packed into the input context window using the custom tokenizer.

After tokenization, the input is fed into BERT, and classification token (CLS) pooling is applied to obtain an embedding representing the entire input context. A dropout layer with dropout rate of 0.1 is used for regularization, and a linear layer maps BERT's 768-dimensional hidden state to the three output classes.

While this layer could be replaced by a more complex MLP, we kept the baseline implementation simple, as we are more interested on evaluating the pre-trained model capabilities. The Single BERT architecture is summarized in Figure 5.

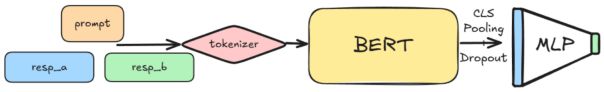

Figure 5. Single BERT Architecture

This setup was implemented using the Transformers library in Python. The model was trained using the Trainer class from the library for both DistilBERT and BERTbase. The evaluation was performed every 500 steps. The learning rate was set to 2e−5, with a per-device batch size of 16 for both training and evaluation. The training was conducted for 3 epochs with a weight decay of 0.01. Mixed precision training (*fp16=True*) was enabled for efficiency. The model was evaluated using the *loss* metric, with *greater_is_better* set to *False*, as lower loss is preferred. To accumulate gradients, *gradient_accumulation_steps=2* was used, and a linear learning rate scheduler with *warmup_steps=500* was applied. Additionally, *load_best_model_at_end=True* ensured that the best model based on evaluation loss was restored at the end of training.

Parallel BERT **(Model 2 /5)**

After designing Single BERT, we recognized that packing the *prompt*, *response_a*, and *response_b* into the 512-token context window likely caused truncation of valuable input information. This truncation may have limited the model's ability to make accurate predictions. To address this, we proposed the Parallel BERT architecture. In our design, the *prompt* and *response_a* are tokenized and fed into BERT, and separately, the *prompt* and *response_b* are tokenized and passed through the same BERT model.

Recall that in Single BERT, all three inputs (*prompt, response_a, and response_b*) were packed into a single input sequence. In contrast, our Parallel BERT setup processes only two segments at a time (*prompt* with *response_a*, and *prompt* with *response_b*), allowing a larger portion of each response to fit within the limited context window and preserving more contextual information.

Furthermore, an important detail in this architecture is that the same BERT model is used for both forward passes, meaning their parameters are shared. This essentially means the only computational cost is the memory during inference time. Following a procedure similar to Single BERT, the outputs are CLS-pooled and passed through a dropout layer. The resulting embeddings are then concatenated into a single vector, which is fed into the classification head to perform the final prediction. The Parallel BERT architecture is summarized in Figure 6.

This was implemented in the Transformers library. The dropout value was 0.1. One notable change from Single BERT is in the size of the output. BERT's hidden state is 768, so

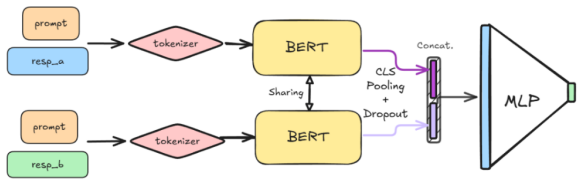

Figure 6. Parallel BERT Architecture

concatenating two outputs results in a 1536-dimensional vector. Consequently, a linear layer of size 1536×3 was used. Another key difference is in the input allocation: using the custom tokenizer, 10% of the context window was allocated to the prompt and 90% to the response. This represents a significant increase in contextual input data compared to the Single BERT approach.

The model was trained using the Trainer class from Transformers for both DistilBERT and BERT-base. The same training parameters as used in the Single BERT model were applied.

Single BERT LLM Data Fine-Tuning **(Model 3/5)**

As highlighted on the Data Augmentation Section, a subset of the training data was expanded into a modified dataset, where each response now has 20 additional labels generated by an LLM across categories such as harmlessness and clarity.

As each response has independent labels, the dataset was restructured so that each row now contains a prompt and a single response, without distinguishing between A and B. This resulted in a modified dataset consisting of 8000 rows.

The goal is to utilize this data to fine-tune the backbone BERT model such that its embedding space aligns more closely with the 20 identified preference categories. To achieve this, a multi-task classification architecture is proposed, enabling BERT's embedding space to shift toward each task-specific representation during fine-tuning. This architecture is illustrated in Figure 7.

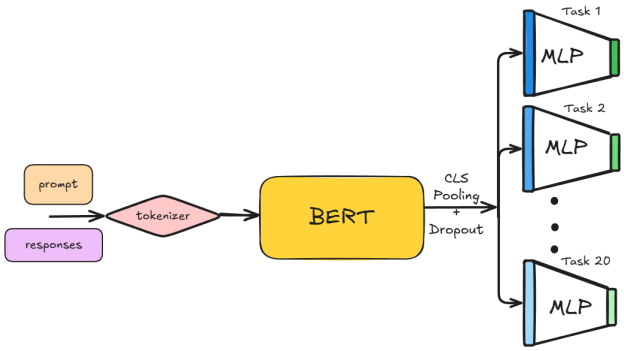

Figure 7. Multi-task Classification BERT Fine-tuning

The input to the architecture is a generic prompt and response pair, which passes through the custom tokenizer (with a prompt and response weighting of [0.1, 0.9]) and into the BERT model. As with previous models, standard CLS pooling and dropout were applied. The resulting embedding was then passed into 20 task-specific heads. Each task head consisted of a single linear layer mapping from BERT's 768-dimensional

hidden space to a size of four, corresponding to the four rating categories assigned by the LLM [0, 1, 2, 3]. The overall loss was computed as the average cross-entropy loss across all 20 tasks. The same training parameters were used as in the Single and Parallel BERT models, with the exception that the number of training epochs was increased to 5 to account for the smaller dataset size.

Following the fine-tuning of BERT on the 20 distinct tasks, the first three encoding layers were frozen to retain some knowledge acquired during this fine-tunning stage. This LLM fine-tuned BERT model was then used for the original classification task of predicting *winner_a*, *winner_b*, or *tie*, using the same Single BERT architecture previously presented. The full model configuration is shown in Figure 8, where the frozen layers are represented with blue shading. This setup was applied using both BERT-base and DistilBERT backbones.

Figure 8. Single BERT LLM Fine-tuned (the blue shading represents the frozen layers)

Parallel Bert LLM Data Fine-Tuning **(Model 4/5)**

Following the same procedure as Single BERT with LLM data fine-tuning, a Parallel BERT version was used in place of the Single BERT architecture. Importantly, this model also leveraged the same BERT backbone fine-tuned on the LLM-generated dataset. The setup was applied using both DistilBERT and BERT-base. This final model is illustrated in Figure 9.

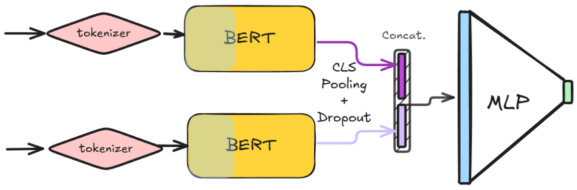

Figure 9. Parallel BERT LLM Fine-tuned (the blue shading represents the frozen layers)

LLama 3.1 8B Model **(Model 5/5)**

As noted in various discussion forums in the Kaggle competition [3], higher-capacity LLMs (decoder-based architectures) appear to perform better on this task. After further investigation, we selected Meta's LLaMA 3.1 model with 8B parameters for our experiments. We believe this model offers a good balance between representational capacity and training complexity.

Classification architecture:

In earlier BERT-based models, classification was straightforward: the model produced an embedding, and an MLP mapped it to one of three target labels. Decoder-based architectures differ—they predict the next token given a tokenized input. For classification, the hidden state of the last non-padding token is extracted and passed through an MLP head to produce class logits. In the case of LLaMA 3.1 8B, this hidden state has a dimensionality of 4096, so the MLP maps from 4096 to 3.

Fine-tuning on the dataset

Despite the "modest" 8B parameter count, full fine-tuning of LLaMA on our setup (2×T4 GPUs, 12 GB VRAM each) is infeasible due to memory and compute constraints. One workaround is freezing layers (layer choice is a hyperparameter). Another more efficient approach is using Low-Rank Adapters (LoRA)[10].

Assume a weight matrix $W_0 \in \mathbb{R}^{M \times N}$ (e.g., an MLP layer without bias). Fine-tuning this directly requires backpropagating through $M \times N$ parameters. Instead, LoRA introduces a trainable low-rank matrix $\Delta W = AB$, where $A \in \mathbb{R}^{M \times d}$ and $B \in \mathbb{R}^{d \times N}$. The modified output becomes:

$$h = W_0 x + \Delta W x = W_0 x + ABx$$

By freezing W_0 and only training A and B, the number of trainable parameters is reduced from $M \times N$ to $M \times d + d \times N$, with d typically ranging from 4 to 64. LoRA — and its quantized variant QLoRA — has been shown to match or even exceed full fine-tuning performance while using a fraction of the resources[10] [11]. The LoRA hyperparameter space is large and includes choices such as the rank d, the scaling factor α, target modules for injection, and dropout rates.

For simplicity and to avoid exhaustive tuning, we used common settings recommended in the literature and community forums: $d = 4$, $\alpha = 8$, with LoRA injected into the `q_proj` and `v_proj` attention layers, following best practices.

Additionally, the token length was limited to 1024 (down from the model's maximum of 4096) to ensure training remained feasible within the Kaggle runtime constraints.

Results

We developed five distinct model architectures, each evaluated using the Kaggle competition test set:

- Model 1/5: Single BERT
- Model 2/5: Parallel BERT
- Model 3/5: Single BERT LLM Data Fine-Tuning
- Model 4/5: Parallel BERT LLM Data Fine-Tuning
- Model 5/5: LLaMA 3.1 8B

Each of the encoder-based models (Models 1–4) was implemented using two backbones: *BERT-base* and *DistilBERT*, resulting in eight encoder configurations. The LLM model was evaluated as a standalone architecture. Test losses are presented in Figure 10, with performance improvements over the baseline (Single DistilBERT) shown in Figure 11.

Examining the model performance data presented in Figures 10 and 11, several key trends emerge that validate our architectural design choices.

First, the parallel architecture consistently outperformed the single architecture across all backbone models. This confirms our hypothesis that processing prompt-response pairs

Figure 10. Model Loss Comparison

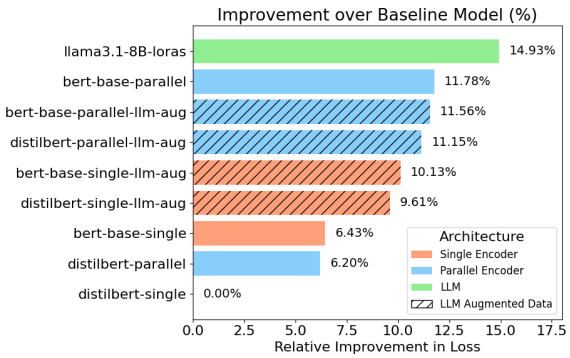

Figure 11. Percentage Improvement over Baseline

separately allows the model to leverage more comprehensive context from both responses, rather than truncating information when all three text segments compete for limited context window space.

Second, our data augmentation strategy using LLM-derived preference annotations proved effective in most configurations. The LLM-augmented DistilBERT models (both single and parallel) showed notable improvements over their non-augmented counterparts. Similarly, the single BERT-base with LLM augmentation outperformed the standard Single BERT-base. However, interestingly, BERT-base Parallel without augmentation slightly outperformed its augmented variant, suggesting that at certain model scales and architectures, the benefits of our augmentation approach may plateau or even introduce noise.

Third, model scaling demonstrates clear benefits for this task. The progression from DistilBERT (66M parameters) to BERT-base (110M parameters) to LLaMA 3.1 (8B parameters) shows a consistent reduction in loss, with LLaMA achieving the lowest overall loss of 0.98. This trend aligns with observations from the competition leaderboard suggesting that higher-capacity models tend to perform better on this task and validating the community findings that that large decoder-only models, despite not being specifically designed for classification tasks, can effectively leverage their extensive pretraining and capacity to excel at human preference modeling.

Discussion and Conclusion

As anticipated, LLaMA 3.1–8B, although not specifically optimized for text classification, achieved the lowest loss on the test set, successfully meeting the performance threshold established for this experiment. This result is primarily attributed to the model's considerable parameter count, which enables it to learn and represent complex patterns within the data. Moreover, LLaMA 3.1–8B supports a significantly larger context window for token input, allowing it to process longer sequences and utilize a broader context when making classification decisions. The combination of high model capacity and extended sequence processing capability appears to have played a crucial role in its superior performance. It is also important to acknowledge that the hyperparameter space explored during testing was constrained due to limited computational resources. We expect that further tuning—particularly with the use of the full 4096-token input length—would likely lead to even better performance.

This trend is also apparent when comparing the results of the encoder-only architectures. Specifically, BERT-base, the largest encoder model in our study, achieved the next best performance when used in conjunction with our parallel approach. BERT-base contains approximately 110 million parameters, a substantial difference from the 8 billion parameters of LLaMA 3.1. When considering the BERT-base single configuration, while the parameter capacity remains constant, the restricted context window (limited to 512 tokens) imposes a significant constraint. As expected, this limitation led to a decline in performance relative to its parallel counterpart.

A similar pattern emerges with DistilBERT. When comparing the single and parallel architectures, the parallel configuration significantly outperforms the single setup. These observations reinforce the hypothesis that context window size plays a crucial role in model performance for this classification task.

Additional insights arise when examining the LLM augmentation strategies, particularly the impact of double fine-tuning on the embedding space. Within the single architecture configurations of both BERT-base and DistilBERT, LLM augmentation yielded improvements in loss reduction. This suggests that fine-tuning may have refined the embedding space toward a richer, task-specific representation, indicating that the embedding space was indeed adjusted to better capture the nuances of the 20 target categories, supporting our initial hypothesis. Nevertheless, this observation warrants further investigation.

One particularly intriguing result emerged when evaluating the LLM augmentation in the parallel architecture. In the case of BERT-base, LLM augmentation did not produce a performance gain, contradicting the trends seen in all previous tests.

Furthermore, we conducted a preliminary investigation into the parameter efficiency of the models. To quantify this, a heuristic metric was devised by taking the inverse of the multiplication of the model's final test loss and its parameter count, under the rationale that both lower loss and fewer parameters are desirable objectives in scenarios of limited computational resources. According to this metric, the LLaMA 3.1-8B model demonstrated a notably low score, indicating that despite its high parameter count, the reduction in loss does not fully justify the increase in complexity. The same metric suggests that the parallel approach offers improved parameter efficiency. Most notably, the highest value of this efficiency metric was found in the LLM-augmented DistilBERT model, further supporting the hypothesis that LLM augmentation effectively enriches the embedding space of the encoder model. Naturally, these findings

warrant further, more rigorous investigation and invite discussion of the tradeoffs of scaling models indefinitely while facing diminishing returns.

In conclusion, this study proposed and evaluated five distinct architectures for the classification task in the LLM Classification Fine-Tuning Kaggle competition (with both smaller and larger backbone models). We successfully met our target objective of achieving a test loss below 1.0, securing a position of 22nd out of 111 on the competition leaderboard. Through this project, we gained valuable insights into the critical role of context window management in transformer-based models for classification tasks, as well as the advantages conferred by employing higher-capacity models. Moreover, by augmenting the training dataset to generate a preference-based dataset using a large LLM, we demonstrated that initial fine-tuning on this enriched dataset effectively shifted the model's embedding space toward a more semantically rich representation. These findings establish a foundation for further research and deeper exploration in this area.

References

1. Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A General Language Assistant as a Laboratory for Alignment. *arXiv e-prints*, page arXiv:2112.00861, December 2021.
2. Junlong Li, Fan Zhou, Shichao Sun, Yikai Zhang, Hai Zhao, and Pengfei Liu. Dissecting Human and LLM Preferences. *arXiv e-prints*, page arXiv:2402.11296, February 2024.
3. Kaggle. Llm classification finetuning. `https://kaggle.com/llm-classification-finetuning`, 2024. Accessed on April 9, 2025.
4. Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2024.
5. LMSYS and SkyLab at UC Berkeley. LMArena: Open Platform for Crowdsourced LLM Evaluation. `https://lmarena.ai/`, 2024. Accessed: 2025-04-13.
6. Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of Draft: Thinking Faster by Writing Less. *arXiv e-prints*, page arXiv:2502.18600, February 2025.
7. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
8. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.
9. Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to Fine-Tune BERT for Text Classification? *arXiv e-prints*, page arXiv:1905.05583, May 2019.
10. Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.
11. Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.